リーダ選出可能な確率的分散アルゴリズムの初期条件 (計算機科学の基礎理論 : 21世紀の計算パラダイムを目指して)

AUTHOR(S):

Sakamoto, Naoshi

CITATION:

Sakamoto, Naoshi. リーダ選出可能な確率的分散アルゴリズムの初期条件 (計算機科学の基礎理論 : 21世紀の計算パラダイムを目指して). 数理解析研究所講究録 2000, 1148: 225-230

ISSUE DATE:

2000-04

URL:

http://hdl.handle.net/2433/63986

数理解析研究所講究録
第1148巻 2000年 225-230

[43] リーダ選出可能な確率的分散アルゴリズムの初期条件

坂本 直志
東京工業大学大学院情報理工学研究科
<sakamoto@noc.titech.ac.jp>

Abstract

In order to elect a leader on anonymous networks by a randomized distributed algorithm, how additional information should be given to the vertices on the networks?

In this paper, by using the notion of "initial condition" introduced by [3, 4], we study the property of the set of the initial condtions with which a randomized algorithm can elect a leader on anonymous networks.

We express that an initial condition A is in $p(N)$-complete set if a randomized distributed algorithm can elect a leader on any anonymous synchronous networks given A with probability at leaset $p(N)$. By our results, $p(N)$-complete set can be classified into four types by $p(N)$ as follows:

1. if $p(N) = 1$, for any initial conditions of 1-complete, there exists a deterministic distributed algorithm to be able to get the number of the vertices on any anonymous networks with the initial information.
2. if the infimum of $p(N)$ is greater than 0, for any intial conditions, there exists a deterministic distributed algorithm to be able to get a value that bounds the size of the network on any anonymous networks with the initial information.
3. if $p(N)$ converges to 0 by increasing N and $\log p(N) = \omega(-N)$, there exists an initial condition of $p(N)$-complete that can't be transformed from and to $\mathcal{UPPERBOUND}$ by any deterministic distributed algorithms.
4. if $\log p(N) = O(-N)$, all initial conditions are $p(N)$-complete.

Moreover, if $p(N)$ converges to 0, the relationship among $p(N)$-complete sets has an infinite hierarchy according to grades of decreasing of $p(N)$.

1 Introduction

One of the most important problem on distributed algorithms on anonymous networks is the leader election problem. This is the problem to make only one computer be a special state by giving the same program to each computer on the network without giving any information like an address.

With respect to this problem, there exists no algorithm solving the leader election problem without assuming any condition. In [2], assuming that each vertex is given the number of the vertices on the network, they showed that there exists a randomized distributed algorithm to give a unique number to each vertex with some probability. On the other hand, in [1, 6], they discussed the class of the graphs where there exist distributed algorithm solving the leader election problem by assuming several conditions.

These results remind the author that an essential information to elect a leader by randomized distributed algorithms might be concerned with the number of the vertices. The notion of view, the set of all paths from a vertex, defined in [6], reminds us to be able to denote the information that determines the behavior of distributed algorithms as the information on the infinite tree. This implies that it is important for designing distributed algorithms calculating in finite time to obtain another additional information to calculate the necessary depth of the view, because each algorithm must finish calculating by using the information of the finite depth of the view. This paper follows that any initial information solving the leader election problem by randomized algorithms contain some information concerned with the size of networks.

For initial condition A and B, If there exists a distributed algorithm such that on any networks satisfying A, the algorithm yields outputs to each vertex satisfying B, then we write the distributed algorithm can transform A to B. We write $A \geq B$ if there exists a distributed algorithm transforming A to B. Then, $\geq$ induces a lattice. In [3, 4], the author showed that there exist infinite initial conditions differed from one another in the sense of $\geq$, among the initial condition having a leader(we denote this initial condition by $\mathcal{LEADER}$), the initial condition having the number of the vertices(we denote this by $\mathcal{SIZE}$) and the initial condition having an upper-bound of the number of the vertices(we denote this by $\mathcal{UPPERBOUND}$). On the other hand, he also showed that for any initial condition A, there exists a deterministic distributed algorithm transforming $\mathcal{LEADER}$ to A.

We investigate the property of initial conditions solving the leader election problem by randomized distributed algorithms with respect to deterministic

distributed algorithms. We write that a initial condition A is $p(N)$-complete if there exists a randomized distributed algorithm transforming A to $\mathcal{LEADER}$ with probability $p(N)$ where N is the size of networks where the algorithm is executed. This is because once a randomized distributed algorithm can transform A to $\mathcal{LEADER}$ with probability $p(N)$, then for any initial condition B, there exists a randomized distributed algorithm transforming A to B with probability $p(N)$ according to the above results. We show there are four cases of the property dependent on $p(N)$ as the following:

1. if $p(N) = 1$:
   for any 1-complete initial condition A, we can design a deterministic distributed algorithm to obtain the number of the vertices from A.

2. if $p(N)$ doesn't converge to 0 by increasing N:
   for any $p(N)$-complete initial condition A, we can design a deterministic distributed algorithm to obtain the upper-bound of the number of the vertices from A.

3. if $p(N)$ converges to 0 by increasing N:
   the following results hold for sufficiently large N.

   - if $\log p(N) = \omega(-N)$:
     there exists an initial condition A such that A is $p(N)$-complete but isn't $q(N)$-complete where $\log q(N) = \omega(\log p(N))$.
   - if $\log p(N) = O(-N)$:
     all initial condition are $p(N)$-complete.

Consider the lattice induced by $\geq$ over the set of $p(N)$-complete initial conditions. Then, its minimum is $\mathcal{SIZE}$ if $p(N) = 1$. On the other hand, its minimum is $\mathcal{UPPERBOUND}$ if $p(N)$ doesn't converge to 0. If $p(N)$ converges to 0, the structure of the lattice of initial conditions depends on the function of the success probability, moreover there is a case where there exists items incomparable with $\mathcal{UPPERBOUND}$.

In Section 2, we define the basic notations. in Section 3, we show the property of the lattice for each success probability. In Section 4, we conclude these.

2 Preliminaries

A network is specified by a graph $G = (V, E, \sigma)$ with a port number, where V is a finite set, $E \subseteq V \times V$, and $\sigma[v]$ $(v \in V)$ is a function that assigns a value from 1 to $\deg(v)$ uniquely to each edge that is connected with v. *The size of* G denotes the number of elements of V. Each vertex represents a processor, and each edge represents a bidirectonal link between the processors of the both ends. A processor v is equipped with $\deg(v)$ input/output ports, and can access to an edge e connected with v by a port number $\sigma[v](e)$.

For a pair of vertices u and v, *path from* u *to* v denotes the row of the vertices $v_0\ v_1 \cdots v_n$ such that v_0 is u, v_n is v, and each v_i and v_{i+1} are adjacent for each $i = 0, \ldots n-1$. The length of this path is n. Let the distance of vertices u and v be the length of the shortest path from u to v. Let the diameter of a network be the maximum length of the distance of all pairs of the vertices in the network.

A *distributed algorithm* is an algorithm given to each vertex (i.e., processor) of a network. We assume that each vertex is given the same algorithm. A network is *anonymous* if each processor v is given no extra information initially except for its local connection, i.e., the number of ports. Below we will introduce a way to give additional information as an initial assignment.

If the port j of v is connected with an adjacent vertex u, then by the instruction "send message m via port j", the message m is sent to u. On the other hand, when v receives some message m, it is also informed the port number j through which m comes from.

An *execution* of a distributed algorithm M is to run M on each vertex. To focus the probabilistic events to only generating random numbers in the process of executing algorithm on each vertex, we discuss about only synchronous networks. That is, we assume that every messages arrives the adjacent vertex in next step.

We investigate the situation where algorithms on a network are given some additional initial information that we call *initial assignment*, or *assignment* for short. An *assignment* is a function from vertices to natural numbers. For a graph G and any assignment a, the graph G with some value on each of its vertices specified by the assignment a is called an *assigned graph*, and it is denoted as G^a. We assume that a distributed algorithm on each vertex on an assigned graph G^a is given the assigned valued of its own vertex.

An *initial condition* is the condition for initial assignments. If a graph is fixed, a set of assignments allowed by a initial condition, is also fixed. Thus, a initial condition is considered as a function from a graph to this set of admissible assignments. In this paper, we assume that the set of admissible assignments for each graph is recursive. Such initial conditions are called *recursive conditions*. Below, we often simply specify an initial condition by giving a way to assign a value to each vertex.

Let us see some examples of initial conditions and initial assignments for a graph of five vertices $G = (\{0, 1, 2, 3, 4\}, E, \sigma)$. For the first example, consider an initial condition that require assignments to give the number of the vertices to all vertices. For our graph G, only one initial assignment satisfies this condition; the assignment a that gives five to all vertices of G, i.e., $a(0) = a(1) = a(2) = a(3) = a(4) = 5$.

Let A be an initial condition that requires assignments to give 1 to one vertex and 0 to the others.

Then for the graph G, the following five initial assignments satisfies A. In other words, $A(G)$ is the set of the following five assignments.

vertex	0	1	2	3	4
a_0	1	0	0	0	0
a_1	0	1	0	0	0
a_2	0	0	1	0	0
a_3	0	0	0	1	0
a_4	0	0	0	0	1

Consider an initial condition B that requires assignments to give a number that upper-bounds the number of vertices to each vertex. Then $B(G)$ is an infinite set because every assignment that gives value larger than five to each vertex satisfies this condition.

While we can consider an initial condition as a function in the above way, we can also consider it as a set of assigned graphs. We write $G^a \in A$ if $a \in A(G)$ for initial condition A.

In this paper, we consider the following initial conditions, which covers almost all initial conditions appeared in the literature.

Definition 2.1

- $\mathcal{LEADER}$: For graph G, $a \in \mathcal{LEADER}(G)$ assigns 1 to one vertex and 0 to the others.
- $\mathcal{SIZE}$: For graph G, $a \in \mathcal{SIZE}(G)$ assigns the number of the vertices of G to all vertices.
- $\mathcal{UPPERBOUND}$: For graph G, $a \in \mathcal{UPPERBOUND}(G)$ assigns a number that bounds the number of vertices of G to all vertices.
- $\mathcal{ZERO}$: For graph G, $a \in \mathcal{ZERO}(G)$ assigns 0 to all vertices.

For initial condition A and B, we express that a distributed algorithm transforms A to B if on any assigned graphs satisfying A, the algorithm yields an assignment satisfying B. $A \geq_{\mathrm{d}} B$ denotes that there exists a deterministic distributed algorithm transforming A to B. It has been shown that $\geq_{\mathrm{d}}$ is a partial order so that it yields a lattice over the equivalent class of the recursive conditions in [3, 4]. Moreover, it also has been shown that $\mathcal{LEADER}$ is a maximum and $\mathcal{ZERO}$ is a minimum in the lattice, and the lattice contains a infinite sequence of equivalent classes, and a infinite equivalence classes such that each pair of them are incomparable each other.

Let $p(N)$ be a function such that on the number N of the vertices it assigns a real number between 0 and 1. If there exists a randomized distributed algorithm transforming A to $\mathcal{LEADER}$ with probability $p(N)$, we write that A is $p(N)$-complete. This is because once a leader is obtained in a network, there exists a deterministic distributed algorithm solving arbitrary initial condition B. Thus we can see that there exists a randomized distributed algorithm such that on any assigned graph satisfying A its output satisfies B with probability $p(N)$.

Proposition 2.2 *If an initial assignment A is $p(N)$-complete, for any initial assignment B, there exists a randomized distributed algorithm such that on any assigned graph satisfying A its output satisfies B with probability $p(N)$.*

We can easily see the following proposition holds according to [2] and [3, 4].

Proposition 2.3 *$\mathcal{SIZE}$ is 1-complete. For any function $p(N)$ such that $0 \leq p(N) < 1$, $\mathcal{UPPERBOUND}$ is $p(N)$-complete.*

Let $p(N)$-complete set be the set of p(N)-complete initial conditions.

Yamashita and Kameda[5] have introduced the notion of "view", which has been important for showing the relation $\geq_{\mathrm{d}}$ not to hold. For any assigned graph G^a and any vertex v of G, the *view* T is a infinite labeled rooted tree that is recursively defined as follows. Its root x, which is labeled as $a(v)$, corresponds to v. It has children $x_1, \ldots, x_{\deg(v)}$ that correspond to the vertices $v_1, \ldots, v_{\deg(v)}$ that are adjacent to v. An edge between x and x_i is labeled as $\sigma[v](v, v_i), \sigma[v_i](v, v_i)$. Then each x_i is a root of the view T.

For a view T, the subtree of T from the root x up to depth i is called a *finite view*, it is denoted as T^i.

Now, we show the following basic property. (The proof is immediate from the definition of view, and it is omitted here.)

Lemma 2.4 *Let G and G' be any networks, and let a and a' be their assignments. If both v of G and v' of G' has the same finite view in depth n in G^a and $G'^{a'}$, then the execution of any deterministic algorithms enters the same state up to n steps.*

Lemma 2.5 *Let G and G' be any networks, and let a and a' be their assignments. If both v of G and v' of G' has the same finite view in depth n in G^a and $G'^{a'}$, then the probability that the execution of any randomized algorithms enters the same state up to n steps is greater than 0.*

3 Results

3.1 In the case where the infimum of the success probability isn't equal to 0:

In this section, we show the property of $p(N)$-complete set where the infimum of $p(N)$ is greater than 0. By designing algorithms actually, we show that for any $p(N)$-complete initial condition A, there exists a deterministic distributed algorithm transforming A to $\mathcal{UPPERBOUND}$, especially there exists the one transforming A to $\mathcal{SIZE}$ if A is 1-complete. First, we define the value τ_q dependent on a assigned graph G^a, a randomized distributed algorithm M and a constant q where $0 \leq q < 1$. Then,

we show the value that bounds this value is computable. Finally, we design deterministic distributed algorithm to obtain the number of the vertices by using this value.

Let G^a be an assigned graph, M be a randomized distributed algorithm, q be a constant where $0 \leq q < 1$, and v be a vertex of G. Let T be the view of v, and T^t be the finite view up to depth t. Let $\mathcal{S}_t$ be the set of assigned graphs whose diameter is larger than t and in which there exists a vertex having the finite view T^t.

Suppose M outputs some value x on the vertex v on G^a in z steps with some probability. Let $pr(H^b)$ be the probability that there exists a vertex on which M yields output x in z steps on H^b. Let π_t be the infimum of this probability for the assigned graphs of $\mathcal{S}_t$. Let τ_q be the greater value between z and the minimum of t so that $\pi_t > q$.

First, we show the property in case of $q = 0$ by the following lemma.

Lemma 3.1 *If $q = 0$, then there exists a deterministic algorithm such that on every input G^a and M, it outputs value τ_0^* where $\tau_0^* \geq \tau_0$.*

Note that this lemma also shows that τ_0 is always finite.

Proof. Suppose that when M is executed on G^a, a vertex v outputs x in z steps with some probability. Then, we consider the finite view T^z with depth z of vertex v. According to Lemma 2.5, if there exists a vertex w that have also T^z on another assigned graph H^b, the probability that w outputs x in z steps isn't equal to 0 when M is executed on H^b. By applying the similar argument to each assigned graph in $\mathcal{S}_z$, we obtain $\pi_z > 0$. Thus $\tau_0 \leq z$. Therefore it satisfies the lemma to compute and output z as follows.

1. Let n be $|G|$. Let i be 1.
2. For every set of n rows $r_1, r_2, \ldots, r_n$ consisting of i bits of 0 and 1, repeat the following steps.
   (a) For each $v_1, v_2, \ldots, v_n$ of G^a, give r_i to v_i respectively as random bits, then simulate M on G^a for i steps.
   (b) if there exists a vertex that outputs some value and halts, output i as the value of z.
3. Increment i by 1, then go back to Step 2.

It is easy to see that this algorithm always outputs some value and terminates in finite steps if the probability that there exists a vertex that outputs some value in finite steps isn't equal to 0 when M is executed on G^a. □

In the definition of τ_0, we considered only the assigned graphs whose diameter is greater than τ_0. But according to Lemma 2.5, we can also prove that for every assigned graph H^b on which there exists a vertex having the finite view T^{τ_0} that is the same finite view as the one on v of assigned graph G^a, the probability is greater than 0 where M is executed for τ_0 steps on H^b.

Now, by using the property in case where $q = 0$, we also show the similar property in case where $q > 0$ holds.

Lemma 3.2 *There exists a deterministic algorithm such that on every input G^a, M and q, it outputs value τ_q^* where $\tau_q^* \geq \tau_q$.*

To prove this lemma, we need the following lemma showing the relation between the diameter and the size of assigned graphs.

Lemma 3.3 *Let $\rho(t,n)$ be $(n-1)((n-2)^t-1)/(n-3)+1$. Let n be the number of the vertices of G^a, and T^t be the finite view of depth t of some vertex of G^a. If there exists a vertex that has the finite view T^t on an assigned graph H^b where its size is greater than $\rho(t,n)$, the diameter of H^b is greater than t.*

The proof of Lemma 3.3 is omitted. Lemma 3.2 can be proved by Lemma 3.1 and Lemma 3.3. And its proof is also omitted.

Next, we design the deterministic algorithms that compute the information concerning the size on $p(N)$-complete initial conditions by τ_q^*.

Theorem 3.4 *If A is 1-complete, $A \geq_{\mathrm{d}} \mathcal{SIZE}$.*

Sketch of proof. Let M be a randomized distributed algorithm transforming A to $\mathcal{SIZE}$ with probability 1.

Let H^b ($\in A$) be an assigned graph where M is being executed. For a vertex v on H^b, we consider the computation of M on v. First, for an integer i, M computes the finite view T^i of v itself.

Suppose that there exists an assigned graph G^a satisfying A and containing a vertex having T^i. If τ_0^* for G^a and M is smaller than i, the probability that there is a vertex on H^b that outputs the same value as an vertex on G^a outputs is not equal to 0. Moreover, the size of H must be equal to the number of the vertices of G by assumption of A and M. Then, in this case, M may output $|G|$ as the size of H^b.

Then, we obtain the size by the following deterministic distributed algorithm:

1. Let i be 1.
2. Obtain the information of the vertices up to distance i from v by communicating with other vertices, then compute the finite view T^i of v by the obtained information.
3. Obtain an assigned graph G^a whose size is smaller or equal to $i+1$ and that containing a vertex having T^i, if such an assigned graph exists in A. Otherwise, increment i by 1, then go back to Step 2.
4. Compute τ_0^* for M and obtained G^a.

5. If $i \geq \tau_0^*$ holds, output $|G|$ and halts. Otherwise, let $i \leftarrow \tau_0^*$ and go back to Step2.

It is easy to see that this algorithm outputs the size correctly if each step works well.

It can be proved that each step can be computed by a deterministic distributed algorithm. □

Theorem 3.5 *For a constant q $(0 < q < 1)$, consider a function $p(N)$ such that $p(N) > 1 - q$ for all N. If A is $p(N)$-complete, $A \geq_{\mathrm{d}} \mathcal{UPPERBOUND}$.*

By the similar way to Theorem 3.4, this theorem can be proved by using τ_q of Lemma 3.2 and $\rho(t, n)$ of Lemma 3.3. But this proof is omitted.

3.2 In the case where the success probability converges to 0:

In this section, we assume that a function $p(N)$ of the success probability is monotone and $\lim_{N\to\infty} p(N) = 0$ holds. On the other hand, we discuss the following for all but finite N, even though we have discussed the theorems holding for all N in the previous section.

That is, in this section, we write that a distributed algorithm transforms A to B if on all but finite assigned graphs satisfying A, it yields an assignment satisfying B, thus, $A \leq_{\mathrm{d}} B$ denotes that there exists a deterministic distributed algorithm transforming A to B in this sense. Hence it follows that $A \not\leq_{\mathrm{d}} B$ denotes that for all deterministic distributed algorithms, on infinitely many assigned graphs satisfying A, the algorithm yields no assignment satisfying B.

Theorem 3.6 *For arbitrary constant c such that $0 < c < 1$, there exists a distributed algorithm to be able to elect a leader with probability c^N for all but finite N.*

Proof. Consider the simple algorithm such that it outputs 1 with probability $1 - c$ and outputs 0 with probability c. If this algorithm is executed on the graph whose size is N, the probability that there is only one vertex outputting 1 is equal to $N(1-c)c^{N-1} = N(1-c)/c \cdot c^N$. This theorem holds whenever $N(1-c)/c \geq 1$ holds. □

Corollary 3.7 *For arbitrary constant c such that $0 < c < 1$, all initial conditions are c^N-complete for all but finite N.*

Definition 3.8 Let $k(N)$-$\mathcal{LEADER}$ be the initial condition such that it assigns 1 to $k(N)$ vertices and 0 to the others for the graphs whose size is N where $1 \leq k(N) \leq N - 1$.

Theorem 3.9 *For any $k(N)$ where $k(N)$ diverges to the infinity by increasing N, $k(N)$-$\mathcal{LEADER}$ and $\mathcal{UPPERBOUND}$ are incomparable under $\leq_{\mathrm{d}}$.*

The proof is omitted.

Theorem 3.10 *If $p(N)$ converges to 0, there exists an initial condition in $p(N)$-complete that is incomparable with $\mathcal{UPPERBOUND}$ under $\leq_{\mathrm{d}}$.*

Sketch of proof. For a constant c such that $0 < c < 1$, we consider $k(N)$ satisfying the following inequation:

$$k(N)(1-c)c^{k(N)-1} \geq p(N).$$

Then it can be proved that $k(N)$-$\mathcal{LEADER}$ is in $p(N)$-complete and incomparable with $\mathcal{UPPERBOUND}$under $\leq_{\mathrm{d}}$. □

Theorem 3.11 *If both $p(N)$ and $p'(N)$ converge to 0 by increasing N where $\log p(N) = \omega(\log p'(N))$ holds, then there exists an initial condition $k(N)$-$\mathcal{LEADER}$ such that there exists a randomized distributed algorithm that can transforming $k(N)$-$\mathcal{LEADER}$ to $\mathcal{LEADER}$ with probability $p'(N)$ but there exists no algorithm that can transform it to $\mathcal{LEADER}$ with probability $p(N)$.*

Sketch of proof. We consider a monotone increasing function $k(N)$ satisfying that

$$p'(N) = k(N)(1-c)c^{k(N)-1}.$$

Then it can be proved that $k(N)$-$\mathcal{LEADER}$ is in $p'(N)$-complete but is not $p(N)$-complete. □

We can see the following lemma by adopting $p(N) = N^{-(\log N)^{i-1}}$ and $p'(N) = N^{-(\log N)^{i-2}}$ where $i \geq 2$ for Theorem 3.11 where $\log p(N) = \omega(\log p'(N))$.

Corollary 3.12 *If $i \geq 2$, for there exists a randomized distributed algorithm transforming $(\log N)^i$-$\mathcal{LEADER}$ to $\mathcal{LEADER}$ with probability $N^{-(\log N)^{i-1}}$, but there exists no randomized distributed algorithm transforming $(\log N)^i$-$\mathcal{LEADER}$ to $\mathcal{LEADER}$ with probability $N^{-(\log N)^{i-2}}$.*

4 Conclusion

We can summarize what we have discussed as follows:

1. $\mathcal{SIZE}$ is a minimum of the lattice of 1-complete.
2. For any $p(N)$ where the infimum of $p(N)$ is greater than 0, $\mathcal{UPPERBOUND}$ is a minimum of the lattice of $p(N)$-complete.
3. For any $p(N)$ where $p(N)$ converges to 0 by increasing N,

(a) if $\log p(N) = \omega(-N)$ holds, for any $q(N)$ where $\log p(N) = \omega(\log q(N))$, there exists an initial condition such that it is $q(N)$-complete, but is not $p(N)$-complete. However, this initial condition is not a minimum of the lattice of $q(N)$-complete, because it is incomparable with $\mathcal{UPPERBOUND}$.

Particularly, $(\log N)^i$-$\mathcal{LEADER}$ is $N^{-(\log N)^{i-1}}$-complete, but is not $N^{-(\log N)^{i-2}}$-complete.

(b) if $\log p(N) = O(-N)$ holds, all initial conditions are $p(N)$-complete.

The problem of the existence of a minimum of the lattice of $p(N)$-complete and the problem what is a necessary and sufficient condition for $p(N)$-complete and $q(N)$-complete to differ, if both $p(N)$ and $q(N)$ converge to 0 are still open.

Acknowledgement

The author would like to thank Prof. Osamu Watanabe for giving useful comments.

References

[1] D. Angluin. Local and global properties in networks of processors. *Proc. 12th ACM Symp. on Theory of Computing*, pp. 82–93, 1980.

[2] J. Bar-Ilan and Dror Zernik. Random leaders and random spanning trees. *Lecture Notes in Computer Science*, Vol. 392, pp. 1–12, 1989. Distributed Algorithms.

[3] Naoshi Sakamoto. On the structure that initial conditions of distributed algorithms make. *IEICE*, Vol. J81–D–I, No. 11, pp. 1190–1200, November 1998.

[4] Naoshi Sakamoto. Comparison of initial conditions for distributed algorithms on anonymous networks. *Proc. 18th Ann ACM Symp. on Principles of Distributed Computing*, pp. 173–179, 1999.

[5] M. Yamashita and T. Kameda. Computing on anonymous networks. *Proc. 7th Ann ACM Symp. on Principles of Distributed Computing*, pp. 117–130, 1988.

[6] M. Yamashita and T. Kameda. Electing a leader when processor identity numbers are not distinct. *Lecture Notes in Computer Science*, Vol. 392, pp. 303–314, 1989. Distributed Algorithms.